실험 4. Key Matrix 구동

1. 목적

임베디드 시스템에서 다수의 스위치 입력이 필요한 경우가 많다. 휴대폰 또는 전화기에 적어도 12개 이상의 스위치가 필요하며, 일반적으로 컴퓨터에 사용하는 키보드의 경우는 100개 이상의 스위치가 필요하다. 이 실험에서는 각 스위치를 한 비트씩 입력 포트에 연결하는 대신, Key Matrix를 구성함으로써 최소한의 입출력 포트를 이용할 수 있도록 구현된 회로를 소프트웨어적으로 이용하는 방법을 배운다.

또 실험에서는 7 Segment LED를 활용한 실습을 통하여 좀 더 실용적인 임베디드 시스템을 구성해 본다.

2. 하드웨어 설명

실험 환경의 가상 스위치 4개 (왼쪽, SW1 ~ SW4) 와 Key Matrix (오른쪽 1,2,3,4 ~ D,E,F,0)는 그림 1과 같으며 회로는 그림 2와 같다.

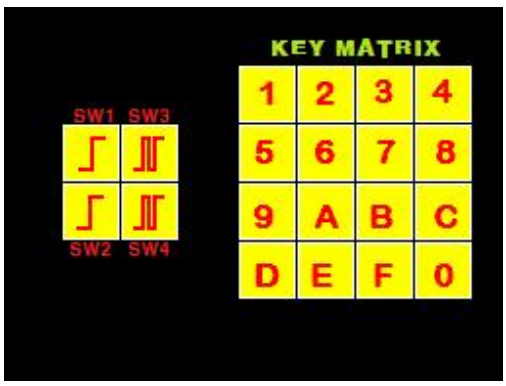


그림 1. 가상 키 그림

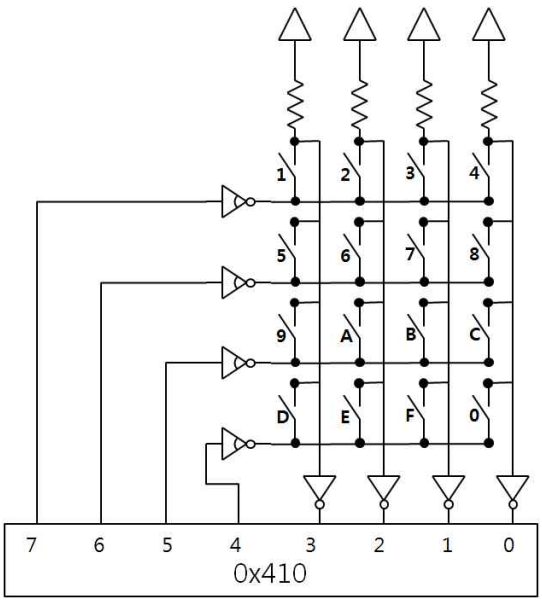


그림 2. Key Matrix 모양과 회로

우선 왼쪽의 Push Switch 들은 Nintendo DS의 실제 Key들과 마찬가지로 readb_virtual_io()를 이용하여 해당 입력 포트 (0x414 번지, PUSH_SW)를 읽음으로써, 해당 스위치를 Bit 단위로 읽을 수 있다. 그림 3과 같이 각 스위치가 비트에 할당되어 있고, 스위치

를 누르면 (즉 화면을 터치하면) 해당 비트가 1이 된다.

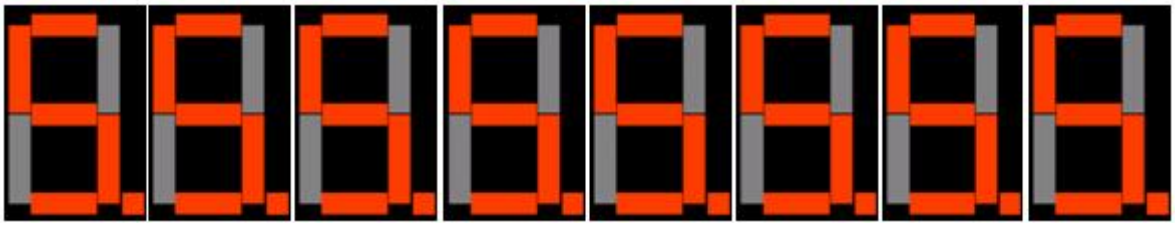
PUSH_SW (0x414 번지)							
7	6	5	4	3	2	1	0
사용하지 않음				SW4	SW3	SW2	SW1

그림 3. 7 Segment LED 구동 출력 포트 구성

Key Matrix는 각 스위치에 하나의 입력 포트를 할당하지 않고, 스위치들을 격자로 배열한 것이다. 그림 1의 Key Matrix는 1...9,A...F,0까지 16개의 스위치를 4 비트의 출력, 4 비트의 입력 비트 만을 가진 회로를 구성하고, 소프트웨어적인 스캔으로 처리한다. 포트(0x410 번지, KEY_MATRIX)의 한 스위치 행에 해당하는 비트 4에서 비트 7을 돌아가면서, 즉, 1234, 5678, 9ABC, DEF0 행 순으로 (역순으로 해도 상관없음), 1을 출력하여 (따라서 해당 Open Collector NOT 게이트에는 0을 출력), 활성화 시키고, 포트의 0에서 3번 비트를 읽는다. 1이 출력된 행의 스위치가 눌리면 해당 열의 포트 입력은 1이 되고, 해당 행의 스위치가 눌리지 않았다면 (다른 행의 스위치가 눌린 것과는 무관하게) 해당 열은 0이 읽힌다. 따라서 순차적으로 각 행을 돌아가며, 어떤 스위치가 눌렸는지 확인이 가능하다. 활성화가 되지 않은 행의 Open Collector NOT 게이트는 마치 회로에 연결되지 않은 듯이 동작하기 때문에, 해당 행의 키를 눌러도 결과에 영향이 없다.

이 방식은 한 번에 한 행의 스위치가 눌려있는지를 확인하는 방식이기 때문에, 문제가 있을 것으로 생각되나, 컴퓨터는 사람이 매우 빠른 속도록 스위치를 누른다 해도, 충분히 빠른 속도로 스캔을 한다면 모든 스위치의 눌림을 빠짐없이 거의 모두 알아낼 수 있으며, 드라이버 소프트웨어를 정교하게 작성함으로써, 여러 키가 동시에 다양한 경우의 키보드 입력을 알아낼 수 있다. PC의 키보드와 같은 다양한 기능이 있는 키보드(Caps-Lock 같은 Lock 키, Shift-Alt-Ctrl과 같은 Modifier Key 등)를 구현하기 위해서는 상당히 복잡한 소프트웨어 구현이 필요하다.

7 Segment LED를 구동하는 SEG7LED(0x400)은 다음과 같은 구성을 가진다. 왼쪽부터 0번 7번까지 8개의 LED가 있고, 각각 0부터 F까지 16개의 숫자를 표시할 수 있다.



LED0 1 2 3 4 5 6 7

그림 4. 7 Segment LED

7 Segment LED를 구동하는 0x400번 가상 출력 포트는 다음과 같이 구성되며, 4, 5, 6번 비트로 켜거나 끌 LED를 선택하고, 켜는 경우, 표시된 숫자를 0에서 3번 비트에 지정한다. 7번 비트를 0으로 하면 LED가 켜지고, 1로하면 꺼진다. 즉 writeb_virtual_io(SEG7LED, 0x??)를 함으로써 7 Segment LED를 구동할 수 있다.

ON/OFF	LED 선택			표시될 숫자 (0..F)			
7	6	5	4	3	2	1	0

그림 6. 7 Segment LED 구동 출력 포트 구성

3. 예제 소스

예제 소스의 함수 Exp_4_Sample_A()는 Key Matrix를 Scan하여 7 Segment LED에 표시하는 프로그램이다.

6, 7번 라인에서 모든 7 Segment LED를 끄고, 9번 라인의 첫 번째 switch 문은 KeyMatrix 왼쪽의 Push SW를 읽기위한 간단한 API인 readb_virtual_io(PUSH_SW)를 보이고 있다. 이렇게 읽은 값을 스위치 번호로 바꾸어, 제일 왼쪽 7 Sgement LED을 켠다.

실제 Key Matrix Scan은 16번 라인에서 scan 변수를 이용하여 행을 지정하고, 18번 라인에서 열을 읽어, 어느 키가 눌렸는지를 행과 열을 이용하여 계산해서 key 변수에 저장하고, 그 값이 16보다 작으면, 29번 라인에서, 해당 키 값을 제일 오른쪽 7 Segment LED에 출력하게 된다. 다른 키 값이 읽힐 때까지 그 값이 제일 오른쪽 7 Segnent LED에 유지된다.

```
01 void
02 Exp_4_Sample_A(void)
03 {
04     u8 key, scan = 0;
05     int i;

06      for (i = 0; i < NUM_7SEG_LED; i++)                // Turn Off All
07        writeb_virtual_io(SEG7LED, 0x80 + (i << 4));

08      while (1) {
09        switch (readb_virtual_io(PUSH_SW)) {
10          case VIRTUAL_SW_1 : writeb_virtual_io(SEG7LED, 0x01); break;
11          case VIRTUAL_SW_2 : writeb_virtual_io(SEG7LED, 0x02); break;
12          case VIRTUAL_SW_3 : writeb_virtual_io(SEG7LED, 0x03); break;
13          case VIRTUAL_SW_4 : writeb_virtual_io(SEG7LED, 0x04); break;
14          default :           writeb_virtual_io(SEG7LED, 0x80); break;
15        }
```

```
16        writeb_virtual_io(KEY_MATRIX, 0x80 >> scan);
17        key = scan * 4;
18        switch (readb_virtual_io(KEY_MATRIX)) {
19          case 8 :  key += 1; break;
20          case 4 :  key += 2; break;
21          case 2 :  key += 3; break;
22          case 1 :  key += 4; if (key == 16) key = 0; break;
23          default : key = 255; break;
24        }
25        scan++;
26        if (scan == 4)
27           scan = 0;
28        if (key < 16)
29           writeb_virtual_io(SEG7LED, 0x70 + key);

30        if (NDS_SWITCH() & KEY_START)
31           break;
32        vTaskDelay(MSEC2TICK(30));
33     }
34     while (NDS_SWITCH() & KEY_START)
35        vTaskDelay(MSEC2TICK(10));       // Wait for releasing START KEY
36 }
```

위 예제 코드를 실행하면, 스캔 방식으로 돌아가며, 각 행을 읽기 때문에, 7 Segment LED가 깜박거리게 된다. (다른 행을 스캔하는 동안은 스위치를 눌러도 반영이 되지 않으므로)

따라서 사람이 누르는 한 번의 동작을 확인하기 위해서는 상태의 관리가 필요하다.

4. 응용 실험 (과제)

이번 과제는 Key Matrix를 구동하여, 누르는 순서대로 7 Segment LED에 누른 키를 제일 왼쪽에 차례로 표시하고, 이전에 누른 키는 왼쪽 방향으로 스크롤 해 나간다. 즉, 7 Segment LED의 제일 왼쪽에는 가장 최근에 누른 키 값이, 제일 오른쪽에서 7번째 전에 눌렀던 키 값이 표시되는 프로그램을 작성하는 것이다.

또, Push Switch 1를 누를 때마다. 7 Segment LED가 Reset 되어 처음 상태 (7 Segment LED에 아무것도 표시가 되지 않은 상태)로 돌아가고, 모드가 전환되어, 가장 최근에 누른 키가 가장 왼쪽 7 Segment LED에 표시되면 오른쪽 방향으로 스크롤 해나가도록 한다.

프로그램은 Template Source의 Exp_Homework.c의 Exp_4_Homework_A() 함수를 수정하여 작성한다.

실험 내용은 꽤 간단하지만, SCAN을 해야하는 Key Matrix를 이용하여 다른 응용을 만드는 것은 결코 쉬운 프로그램이 아니다. 다음 실험을 위해서 Key Matrix를 읽는 일반적인 함수를 만들어 놓는 것이 바람직하다.

5-1. 소스 및 샘플 프로그램 (게시판 이용)

A. Template 소스는 자료실의 KeyMatrix.zip에 있음
B. 미리 만들어 놓은 Binary는 자료실의 KeyMatrix.nds 임

5-2. 소스 및 샘플 프로그램 (Github 이용)

소스는 Github의 KeyMatrix 소스를 check-out하여 과제를 수행한다.

https://github.com/hl1itj/nds-ide/tree/microprocessor/lab/keymatrix

(이 문서와 미리 build한 binary KeyMatrix.nds는 doc directory에 있음)

Checkout 한 뒤,
https://github.com/hl1itj/nds-ide/tree/microprocessor/학번/keymatrix

에서 각자 개발을 진행한다.